

CLAIMS

What is claimed is:

- 1 1. A method for shadow mapping while rendering a primitive in a graphics
2 pipeline, comprising:
3 (a) performing an offset operation to generate a depth value while rendering a
4 primitive;
5 (b) identifying a value of a slope associated with a primitive; and
6 (c) conditionally clamping the depth value based on the value of the slope.
- 1 2. The method as recited in claim 1, wherein the shadow mapping process
2 includes rendering the primitive from a light space perspective.
- 1 3. The method as recited in claim 1, wherein the depth value is clamped if the
2 value of the slope is greater than a predetermined amount.
- 1 4. The method as recited in claim 1, wherein the clamping includes the steps of:
2 identifying vertex depth values of vertices of the primitive; comparing at
3 least one of the vertex depth values with the depth value generated by the
4 offset operation; and clamping the depth value generated by the offset
5 operation based on the comparison.
- 1 5. The method as recited in claim 4, wherein the depth value generated by the
2 offset operation is clamped to the depth value generated by the offset
3 operation if the depth value generated by the offset operation is less than the
4 maximum vertex depth value.
- 1 6. The method as recited in claim 4, wherein the depth value generated by the
2 offset operation is clamped to the greatest vertex depth value if the greatest

3 vertex depth value is less than the depth value generated by the offset
4 operation.

1 7. The method as recited in claim 4, wherein the depth value generated by the
2 offset operation is clamped to the depth value generated by the offset
3 operation if the depth value generated by the offset operation is greater than
4 the least one of the vertex depth values.

1 8. The method as recited in claim 4, wherein the depth value generated by the
2 offset operation is clamped to the least one of the vertex depth values if the
3 least one of the vertex depth values is greater than the depth value generated
4 by the offset operation.

1 9. The method as recited in claim 1, wherein the offset operation includes a
2 polygon offset operation in accordance with the OpenGL[®] programming
3 language.

1 10. A computer program embodied on a computer readable medium for shadow
2 mapping while rendering a primitive in a graphics pipeline, comprising:
3 (a) a code segment for performing an offset operation to generate a depth value
4 while rendering a primitive;
5 (b) a code segment for identifying a value of a slope associated with a slope of
6 the primitive; and
7 (c) a code segment for conditionally clamping the depth value based on the value
8 of the slope.

1 11. The computer program as recited in claim 10, wherein the depth value is
2 clamped if the value of the slope is greater than a predetermined amount.

1 12. The computer program as recited in claim 10, wherein the clamping includes:
2 identifying vertex depth values of vertices of the primitive; comparing at

3 least one of the vertex depth values with the depth value generated by the
4 offset operation; and clamping the depth value generated by the offset
5 operation based on the comparison.

1 13. The computer program as recited in claim 12, wherein the depth value
2 generated by the offset operation is clamped to the depth value generated by
3 the offset operation if the depth value generated by the offset operation is less
4 than the maximum vertex depth value, and wherein the depth value generated
5 by the offset operation is clamped to the greatest vertex depth value if the
6 greatest vertex depth value is less than the depth value generated by the
7 offset operation.

1 14. The computer program as recited in claim 12, wherein the depth value
2 generated by the offset operation is clamped to the depth value generated by
3 the offset operation if the depth value generated by the offset operation is
4 greater than the least one of the vertex depth values, and wherein the depth
5 value generated by the offset operation is clamped to the least one of the
6 vertex depth values if the least one of the vertex depth values is greater than
7 the depth value generated by the offset operation.

1 15. A system for shadow mapping while rendering a primitive in a graphics
2 pipeline, comprising:
3 (a) logic for performing an offset operation to generate a depth value while
4 rendering a primitive;
5 (b) logic for calculating and identifying a value of a slope associated with the
6 primitive; and
7 (c) logic for conditionally clamping the depth value based on the value of the
8 slope.

1 16. A method for performing shading calculations in a graphics pipeline,
2 comprising:

- 3 (a) performing a first shading calculation in order to generate output;
- 4 (b) saving the output; and
- 5 (c) performing a second shading calculation using the output in order to generate
- 6 further output.

1 17. The method as recited in claim 16, wherein the first shading calculation
2 includes $[(1-s) * (\text{Color_diff} + \text{Color_spec})]$ for generating an output A, and
3 the second shading calculation includes $[\text{Color_amb} + A]$, where s is a
4 shadow variable, Color_diff is a diffuse color variable, Color_spec is a
5 specular color variable, and Color_amb is an ambient color variable.

1 18. The method as recited in claim 16, wherein the first shading calculation
2 includes $[(1-s) * \text{Color_diff} + \text{Color_amb}]$ for generating an output A, and
3 the second shading calculation includes $[A * \text{Texture_det} + (1-s) * \text{Color_spec}]$, where s is a shadow variable, Color_diff is a diffuse color
4 variable, Color_spec is a specular color variable, Color_amb is an ambient
5 color variable, and Texture_det is a detail texture variable.

1 19. The method as recited in claim 16, wherein the first and second shading
2 calculations together include a diffuse color variable, a specular color
3 variable, and an ambient color variable.

1 20. The method as recited in claim 19, wherein the variables are decoupled.

1 21. The method as recited in claim 16, wherein the method is carried out with a
2 system comprising:

- 3 (a) a shading module for performing the first shading calculation in order to
4 generate the output;
- 5 (b) a texture look-up module coupled to the shading module for retrieving
6 texture information using texture coordinates associated with the output;

- 7 (c) a feedback loop coupled between an input and an output of the shading
8 module for performing the second shading calculation using the texture
9 information from the texture look-up module in order to generate further
10 output; and
11 (d) a combiner module coupled to the output of the shading module for
12 combining the output generated by the shading module.

- 1 22. A computer program embodied on a computer readable medium for
2 performing shading calculations in a graphics pipeline, comprising:
3 (a) a code segment for performing a first shading calculation in order to generate
4 output;
5 (b) a code segment for saving the output; and
6 (c) a code segment for performing a second shading calculation using the output
7 in order to generate further output.

- 1 23. The computer program as recited in claim 22, wherein the first shading
2 calculation includes $[(1-s) * (\text{Color_diff} + \text{Color_spec})]$ for generating an
3 output A, and the second shading calculation includes $[\text{Color_amb} + A]$,
4 where s is a shadow variable, Color_diff is a diffuse color variable,
5 Color_spec is a specular color variable, and Color_amb is an ambient color
6 variable.

- 1 24. The computer program as recited in claim 22, wherein the first shading
2 calculation includes $[(1-s) * \text{Color_diff}] + \text{Color_amb}$ for generating an
3 output A, and the second shading calculation includes $[A * \text{Texture_det} + (1-$
4 $s) * \text{Color_spec}]$, where s is a shadow variable, Color_diff is a diffuse color
5 variable, Color_spec is a specular color variable, Color_amb is an ambient
6 color variable, and Texture_det is a texture detail variable.

- 1 25. The computer program as recited in claim 22, wherein the first and second
2 shading calculations together include a diffuse color variable, a specular
3 color variable, and an ambient color variable.
- 1 26. The computer program as recited in claim 25, wherein the variables are
2 decoupled.
- 1 27. The computer program as recited in claim 22, wherein the code segments are
2 carried out with a system comprising:
3 (a) a shading module for performing the first shading calculation in order to
4 generate the output;
5 (b) a texture look-up module coupled to the shading module for retrieving
6 texture information using texture coordinates associated with the output;
7 (c) a feedback loop coupled between an input and an output of the shading
8 module for performing the second shading calculation using the texture
9 information from the texture look-up module in order to generate further
10 output; and
11 (d) a combiner module coupled to the output of the shading module for
12 combining the output generated by the shading module.
- 1 28. A system for performing shading calculations in a graphics pipeline,
2 comprising:
3 (a) logic for performing a first shading calculation in order to generate output;
4 (b) logic for saving the output; and
5 (c) logic for performing a second shading calculation using the output in order to
6 generate further output.